Spring 5-20-2019

Malware Analysis on PDF

Shubham Shashishekhar Pachpute
San Jose State University

Malware Analysis on PDF

A Project

Presented to

The Faculty of Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Shubham Shashishekhar Pachpute

March 2019

The Designated Project Committee Approves the Project Titled

Malware Analysis on PDF

By

Shubham Shashishekhar Pachpute

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

San José State University

May 2019

Dr Robert Chun	Department of Computer Science
Dr Mike Wu	Department of Computer Science
Mr Navneet Goel	FireEye Inc

ABSTRACT

Cyber-attacks are growing day by day and attackers are finding new techniques to cause harm to their target by spreading worms and malware. In the world of innovations and new technologies coming out every day, it creates a possibility of attacking a system and exploiting the vulnerabilities present in the system. One of the methods used for the spread of malware is the Portable Document Format (PDF) files. Due to the flexible nature of these files, it is becoming a sweet spot for the attackers to embed the malware easily into the PDF files. In this report, we are going to understand this flexibility provided by the PDF development and why the bad actors find it easy to embed viruses or malware into the PDF files. We will then look at how we can develop methods and techniques using python script to identify the malicious files and stop it from harming the systems in your network.

Keywords – cyber-attacks, malware, Portable Document Format (PDF).

ACKNOWLEDGEMENTS

I am grateful to take this opportunity to sincerely thank my project advisor, Dr. Robert Chun, for his constant support, invaluable guidance, and encouragement. His work ethic and constant endeavor to achieve perfection have been a great source of inspiration.

I wish to extend my sincere thanks to Dr. Mike Wu and Mr. Navneet Goel for consenting to be on my defense committee and for providing invaluable suggestions to my project without which this project would not have been successful.

I also would like to thank my friends and family, for their constant support and encouragement throughout my graduation and for believing in me.

TABLE OF CONTENTS

List of Figures

Chapter 1

Introduction

In recent years, there has been a huge increase in the services that are provided on the Internet. With the use of smartphones and powerful computers, the power to do any task is readily available. When e-mail was introduced, people not only started sending messages anywhere in the globe, but also it enabled services such as sharing files over e-mail. When the memory on the physical devices could not suffice the needs, cloud storage was introduced which allowed unlimited storage to anyone, anywhere with an access to internet.

However, these technological advancements have also brought in severe drawbacks. The amount of data that is available on the internet can be a possible target of a cyber-attack which can leak a person's personal information and cause him serious damage. Cybercrime is on the rise and has become a profitable activity for the bad actors in the black market. For example, recently, we have seen a lot of malware attacks causing damage to governments, private sectors and the healthcare industry. These attackers are economically motivated which leads to development and evolution of these malware which are sustaining over the web.

Portable Document Format (PDF) files have become an added challenge to stop the spreading of malware over the internet. The ability of including different types of content with embedded JavaScript and ActionScript codes makes it an easy target for malware attacks. Adobe, who came up with the PDF document type in the early 1990s, still publicly discloses some critical vulnerabilities in the PDF documents with vulnerabilities in the acrobat reader software which compiles these files, leaving behind serious potential threats for its users as mentioned in the research work [1], [2].

Due to the flexibility provided by the PDF format, it has been a challenge to detect the malicious PDF files and often troubles the forensic analyst to find and hunt these files. Machine learning has been seen as a new tool to tackle such complex tasks in forensic investigation and cyber security [3]. But, to develop an efficient machine learning model, it is foremost important to identify the features which will help classify the files as malicious or benign. Machine learning was not originally developed for such investigations and therefore, it leaves some vulnerabilities behind which can be used by the attackers for their actions. Nevertheless, machine learning can benefit the forensic analyst and can greatly affect the detection process and help curb down these attacks.

In this report, we are going to see, what is a PDF file? Why is it easy to embed malware in a PDF document? How is it so effective? And how can we stop the attacks carried out using the PDF documents?

In this report, Section II explores the structure of the PDF document and gives insights into understanding how the PDF document is coded. In Section III we will explore the malware that are seen in the PDF documents and how they exploit the systems by exploiting the vulnerabilities in the PDF readers. Section IV talks about the various techniques and methods that can be used to find malicious PDF documents. Figure 1 shows the organization of the report and the topics it will cover.

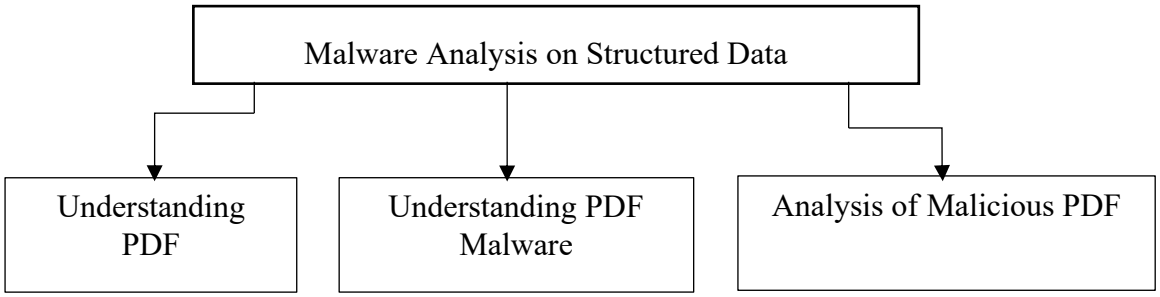

Fig 1. Organization of the report

Chapter 2

Understanding PDF

In this chapter, we are going to talk about the PDF file and understand its structure which is a stepping stone of understanding how a malware is embedded into the files or how we can identify the abnormalities in the given PDF file structure.

PDF was invented by Adobe Systems in 1993. It supported text, images, hypertext links, bookmarks and thumbnails preserving them in their original formatting and appearance. The first revision of PDF in 1994 supported adding passwords to the documents and encryption to add the security feature with its flexibility in type of data that can be shared. Over the years, we have seen a huge increase in the use of PDF formats. Today, it supports 3D artwork, editable forms and animations which can be rendered in the PDF. AES encryptions have also been introduced for better data security and integrity. It also supports technologies such as JavaScript and ActionScript for visualizing text.

Let us now understand the syntax of a PDF document.

- Objects: The document is a data structure which is constructed from a collection of different types of data objects. There are two types of conventions used to write these objects:
    - Lexical Conventions: set of character used to write the syntactic elements and objects.
    - Stream Objects: explains the writing of complex data type, the stream objects. It is a sequence of bytes comprising of a dictionary and the keywords **stream** and **endstream** within which the byte sequence is enclosed. Stream objects are indirect objects. Figure

2 below shows an example of how a PDF stream object is represented and the various components it contains.

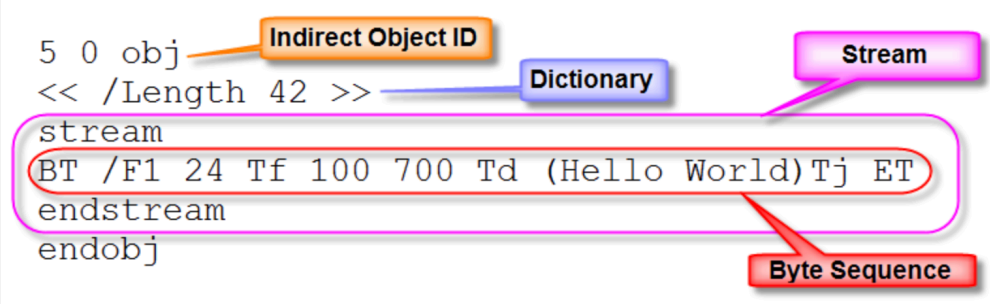

Fig 2. Example of PDF Stream Object [27]

- File Structure: It helps determine how the objects in the PDF document are stored, accessed and updated.
    - A PDF document is a collection of objects which contains a single self-contained sequence of bytes with the associated structural information. Samir G. Sayed and Mohamed Shawkey in [8] talk about the structure of a PDF document. Fig 3. shows the typical structure of a PDF document. A PDF file comprises of four components namely: header, objects, cross reference (xref), and trailer.
    - The header provides information about the version of the PDF language. The header is included in the beginning of a PDF file. A PDF renderer ignores the file if its header is missing.
    - The body which consists of one or more objects. There are several types of objects such as strings, numbers, dictionaries, boolean, and streams. Objects have information including fonts, graphics, pages, and embedded codes such as JavaScript and Acrobat forms. Dictionary object contains simpler types of objects such as names, arrays, and numbers. Streams are used to store a large amount of embedded data such as images, multimedia, fonts, and JavaScript. Streams in benign PDF files normally contain text,

fonts, and pictures. However, streams in malicious PDF files often contain JavaScript. Attackers use compressed streams to hide the malicious JavaScript codes, otherwise these malicious codes will be plain text and easily visible.

- The cross-reference (xref) table contains list of all objects and offsets to the beginning of these objects in the PDF file. The trailer of the PDF file provides information about the offset to the location of the cross-reference table and objects.

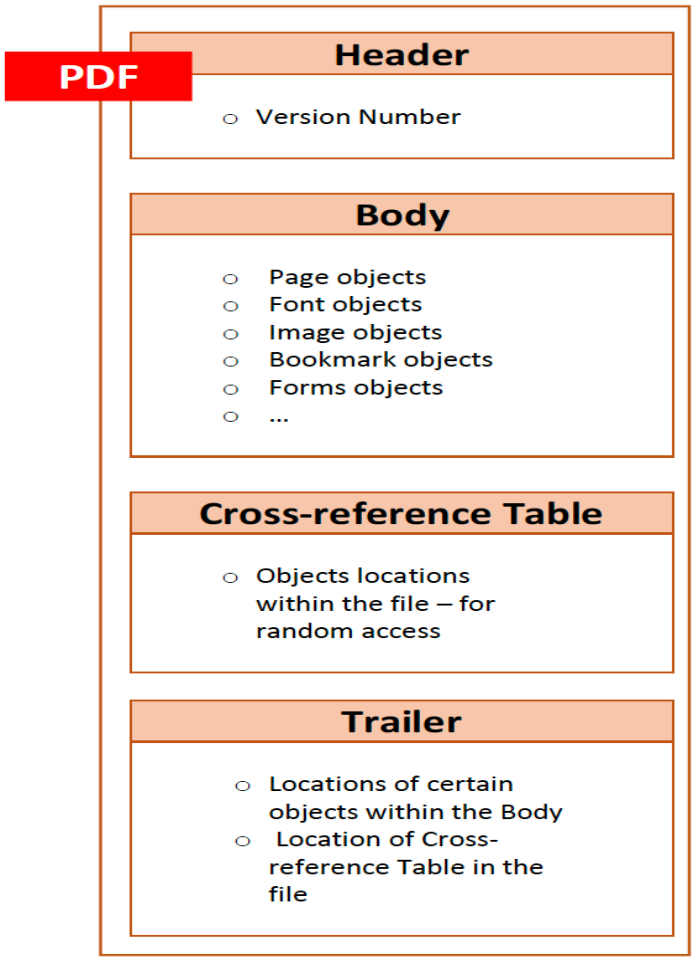

Fig 3. PDF file structure [28]

- When a reader starts to process a PDF, it begins from the trailer object i.e. by finding the */Root* and then constructs the indirect object and continues to decompress the data. Fig 4. shows a typical example of how a PDF document generated looks.

```
%PDF-1.1 Header
1 0 obj Top-level of page tree: has two children—page one and an intermediate page tree node
<< /Kids [2 0 R 3 0 R] /Type /Pages /Count 3 >>
endobj
4 0 obj Contents stream for page one
<< >>
stream
1. 0.000000 0.000000 1. 50. 770. cm BT /F0 36. Tf (Page One) Tj ET
endstream
endobj
2 0 obj Page one
<<
  /Rotate 0
  /Parent 1 0 R
  /Resources
    << /Font << /F0 << /BaseFont /Times-Italic /Subtype /Type1 /Type /Font >> >> >>
  /MediaBox [0.000000 0.000000 595.275590551 841.88976378]
  /Type /Page
  /Contents [4 0 R]
>>
endobj
5 0 obj Document catalog
<< /PageLayout /TwoColumnLeft /Pages 1 0 R /Type /Catalog >>
endobj
10 0 obj Document information dictionary
<<
  /Title (PDF Explained Example)
  /Author (John Whitington)
  /Producer (Manually Created)
  /ModDate (D:20110313002346Z)
  /CreationDate (D:2011)
>>
endobj xref
0 11
trailer Trailer dictionary
<<
  /Info 10 0 R
  /Root 5 0 R
  /Size 11
  /ID [<75ff22189ceac848dfa2afec93deee03> <75ff22189ceac848dfa2afec93deee03>]
>>
startxref
0
%%EOF
```

Fig 4. Example of a PDF document

- Incremental Updates: A PDF file can be updated by adding the new content to the end of the file and keeping the original content as it is. When a file is updated, a cross

reference section is also added which contains information of only the newly added objects or the objects that were changed or deleted.

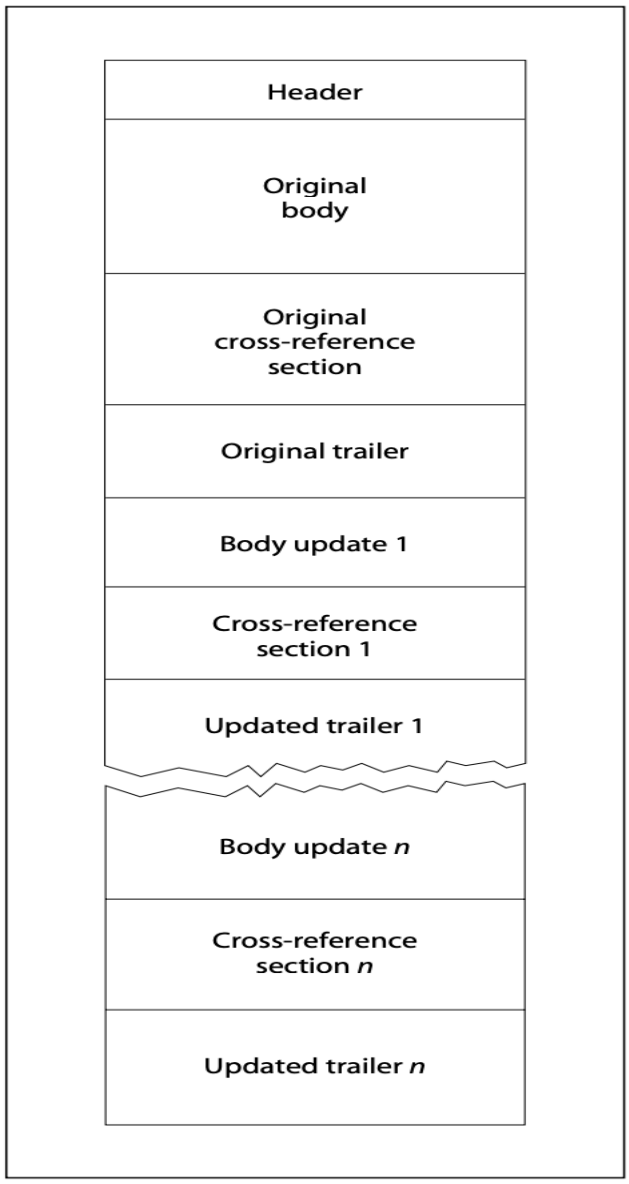

Fig 5. Structure of an updated PDF file. [29]

- Document Structure: It describes the representation of the basic object types and the components of a PDF documents.

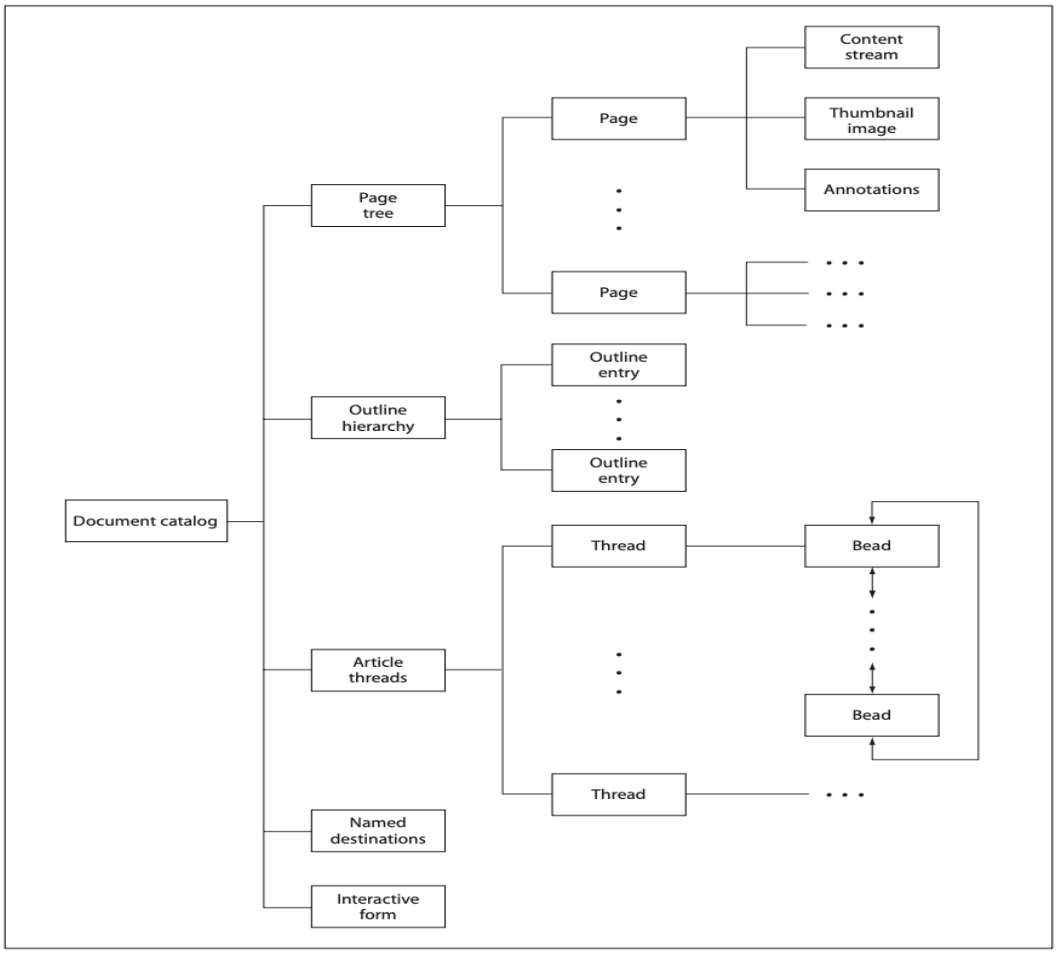

Fig 6. Tree Structure of a PDF document [29]

- The root of the document is located in the trailer section of the document. The root is a catalog dictionary which defines information of the content, article threads, outline, and named destinations. It also contains information of how the data would look in the document. The document follows a page tree structure in maintaining the entire content which also allows various applications to quickly travel through 1000s of pages and fetch the content to display. Intermediate nodes called page tree nodes and leaf nodes as page objects are the two different types of objects in the tree structure. Acrobat Distiller and PDF Writer programs construct tress which are known as balanced trees.

Chapter 3

Understanding the PDF Malware

As we have seen in the last section, a PDF document allows for different kinds of objects to be embedded to provide a suitable way of sharing the information and content a user need. Due to these options, it provides a possibility for the attackers to exploit these features of the document by embedding malicious codes or malware into the PDF. With such flexibility in the document itself, it becomes easier for the attackers to exploit the software which reads this document. Davide Maiorca and Battista Biggio talk about the examples of malware embedded in a PDF document. Common Vulnerabilities and Exposures (CVE) report for adobe reader over the years shows the vulnerability report of the software which could be exploited by the attackers by infusing the malicious codes in the document. Figure 7 shows the CVE report for Adobe Reader over the years and the different types of vulnerabilities are can be exploited.

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score
1	CVE-2018-19722	125			2019-01-18	2019-01-23	5.0
Adobe Acrobat and Reader versions 2018.011.20063 and earlier, 2017.011.30102 and earlier, and 2015.006.30452 information disclosure.							
2	CVE-2018-19719	125			2019-01-18	2019-01-22	4.3
Adobe Acrobat and Reader versions 2019.008.20081 and earlier, 2019.008.20080 and earlier, 2019.008.20081 and and earlier, and 2015.006.30456 and earlier have an out-of-bounds read vulnerability. Successful exploitation could							
3	CVE-2018-19717	125			2019-01-18	2019-01-23	4.3
Adobe Acrobat and Reader versions 2019.008.20081 and earlier, 2019.008.20080 and earlier, 2019.008.20081 and and earlier, and 2015.006.30456 and earlier have an out-of-bounds read vulnerability. Successful exploitation could							
4	CVE-2018-19716	119		Exec Code Overflow	2019-01-18	2019-01-23	7.5
Adobe Acrobat and Reader versions 2019.008.20081 and earlier, 2019.008.20080 and earlier, 2019.008.20081 and and earlier, and 2015.006.30456 and earlier have a heap overflow vulnerability. Successful exploitation could lead t							
5	CVE-2018-19715	416		Exec Code	2019-01-18	2019-01-23	10.0
Adobe Acrobat and Reader versions 2019.008.20081 and earlier, 2019.008.20080 and earlier, 2019.008.20081 and and earlier, and 2015.006.30456 and earlier have a use after free vulnerability. Successful exploitation could lead tc							
6	CVE-2018-19714	125			2019-01-18	2019-01-22	4.3
Adobe Acrobat and Reader versions 2019.008.20081 and earlier, 2019.008.20080 and earlier, 2019.008.20081 and and earlier, and 2015.006.30456 and earlier have an out-of-bounds read vulnerability. Successful exploitation could							
7	CVE-2018-19713	416		Exec Code	2019-01-18	2019-01-23	9.3
Adobe Acrobat and Reader versions 2019.008.20081 and earlier, 2019.008.20080 and earlier, 2019.008.20081 and and earlier, and 2015.006.30456 and earlier have a use after free vulnerability. Successful exploitation could lead tc							

Fig 7. Known CVE on PDF Document

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
1999	1		1	1											
2000	1		1	1											
2001	1														
2002	1														
2003	3		2	1											
2004	6		5	4											
2005	9	4	5	3											
2006	7	2	3		1							2			
2007	9	3	3		1		2		1				1		1
2008	17	2	12	4	2							1			
2009	45	15	36	20	13					1		1			3
2010	68	35	60	33	29		2			3		1			4
2011	60	21	48	33	17		3			2		6			1
2012	30	24	30	24	23					1					
2013	66	30	60	49	30					3	1	1			
2014	44	17	35	17	17		1			5	4				
2015	137	29	61	39	24					61	20				
2016	21	11	18	11	11					1		2			
2017	130		82	54	56					6	35				
2018	44		16	4	1					1	1				
Total	700	193	478	298	225		8		1	84	61	14	1		9
% Of All		27.6	68.3	42.6	32.1	0.0	1.1	0.0	0.1	12.0	8.7	2.0	0.1	0.0	

Fig 8. CVE report for Adobe reader

JavaScript code, encoded streams and embedded objects are first choices of the attackers to exploit these vulnerabilities. It allows the execution of a remote code or create remote network connections as soon as the document is opened by the reader. One of the examples that exploits Adobe reader allowed downloading and installing a malware from a remote server. This was a BMP/RLE heap corruption vulnerability (CVE-2013-2729) which allows the heap to overflow and execute a remote connection code [3]. One more example of vulnerability in Adobe Reader is about the EScript.api which allows remote attackers run an arbitrary code or a denial of service (DoS) attack on the target system given in (CVE-2010-4091).

Davide Maiorca and Battista Biggio in [3] and Soon Heng Tan Mavric and Chai Kiat Yeo in [10] have both given examples on Javascript code embeeded in the document. CVE-2010-1240

is a vulnerability which executes binary code simply when the PDF document is opened [3]. This was reported by Contagio in 2010. The sample of the code embedded is as follows [3]:

```
<<
/Type /Action
/S /Launch
/Win
<<
/F (cmd.exe)
  /P (/c echo Dim BinaryStream > vbs1.vbs && echo Set BinaryStream =
CreateObject("ADODB.Stream")
>>
....
Endobj
```

When the reader opens the document, the file executed a command in the Windows Shell which is a Visual Basic script. This vulnerability was used to execute the Zeus Trojan which is a Trojan horse malware, generally used to steal private information such as banking records by keystroke logging.

The above-mentioned examples only compose a handful of malicious activities and vulnerabilities that can harness over the PDF document platform and expose the readers used for rendering these documents for user readability purposes. Thus, these attacks can get very sophisticated and complex. Identifying and finding them every time is a critical task.

Let us look at some of the PDF tags which can be risky and cause damage:

- /OpenAction and /AA: It will execute a task or an action automatically.
- /JavaScript and /JS specify the script to run which can execute a script or open a backdoor.
- /GoTo changes the view to a specific destination within the document. It could also point to another document.
- /Launch can be used to open a document or start/initiate a program.
- /URI will fetch and access a URL.
- /SubmitForm and /GoToR posts data to the given url.
- /RichMedia: embeds Flash in the document.
- /ObjStm: Hides an Object Stream.

Chapter 4

Experiments and Results:

Analysis of Malicious PDF

We have seen how attacks can be made possible via the PDF document by embedding code into it. This helps exploit the reader and target systems. To avoid this, there are various analysis methods which could be used to identify malicious PDF documents from the non-malicious ones. These processes can be difficult to implement and consume time, but with the growing attacks and their consequences, it becomes more and more important to invest this time and find the documents produced by the attackers. We have collected a dataset of over 24000 files comprising of 9000 clean PDF files and 15000 Malicious PDF files collected over the internet.

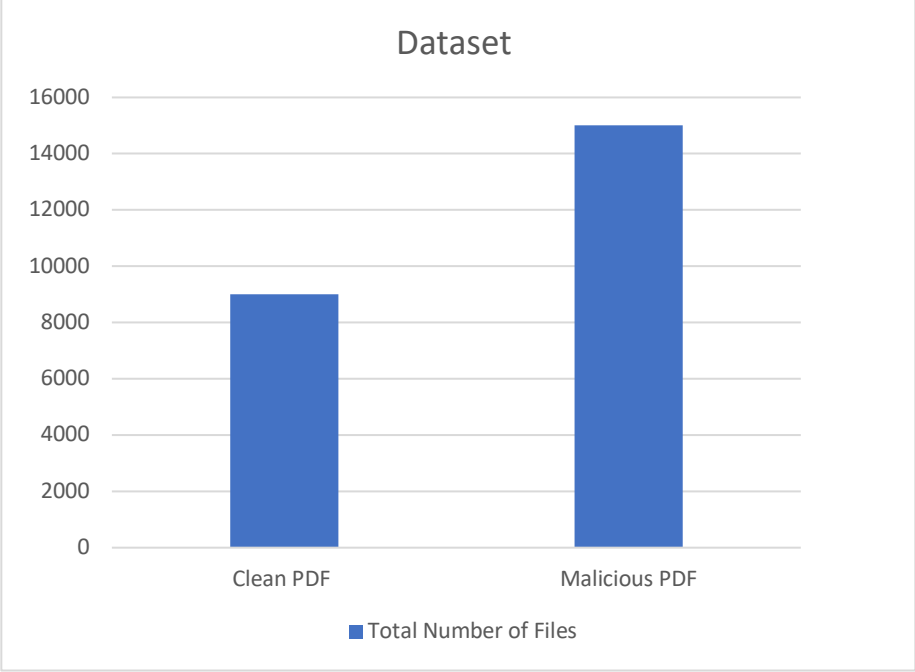

Fig 9. Data Set

Let us study one of the open source tools which is widely used for pdf analysis.

- peepdf:

  It is an open source python tool which helps to analyze a given PDF file. It can help a security researcher give valuable information and acquire all the necessary objects required by the researcher.

  Usage:

```
Shubhams-MacBook-Pro:peepdf_0.3 pachputeshubham$ python peepdf.py
Usage: peepdf.py [options] PDF_file

Version: peepdf 0.3 r235

Options:
  -h, --help            show this help message and exit
  -i, --interactive     Sets console mode.
  -s SCRIPTFILE, --load-script=SCRIPTFILE
                        Loads the commands stored in the specified file and
                        execute them.
  -c, --check-vt        Checks the hash of the PDF file on VirusTotal.
  -f, --force-mode      Sets force parsing mode to ignore errors.
  -l, --loose-mode      Sets loose parsing mode to catch malformed objects.
  -m, --manual-analysis
                        Avoids automatic Javascript analysis. Useful with
                        eternal loops like heap spraying.
  -u, --update          Updates peepdf with the latest files from the
                        repository.
  -g, --grinch-mode     Avoids colorized output in the interactive console.
  -v, --version         Shows program's version number.
  -x, --xml             Shows the document information in XML format.
Shubhams-MacBook-Pro:peepdf_0.3 pachputeshubham$
```

Fig 10. Usage of peepdf tool

It provides multiple options which can help a researcher dig deep and understand the structure and contents of the file.

```
Shubhams-MacBook-Pro:peepdf_0.3 pachputeshubham$ python peepdf.py -i
PPDF> help

Documented commands (type help <topic>):
========================================
bytes           exit         js_join          quit          set
changelog       filters      js_unescape      rawobject     show
create          hash         js_vars          rawstream     stream
decode          help         log              references    tree
decrypt         info         malformed_output replace       vtcheck
embed           js_analyse   metadata         reset         xor
encode          js_beautify  modify           save          xor_search
encode_strings  js_code      object           save_version
encrypt         js_eval      offsets          sctest
errors          js_jjdecode  open             search

PPDF>
```

Fig 11. Tools provided by peepdf

```
PPDF> metadata

Info Object in version 0:

<< /ModDate D:20091004134555
/CreationDate D:20091004134555
/Producer Scribus PDF Library 1.3.5svn
/Creator Scribus 1.3.5svn
/Title
/Trapped /False
/Keywords
/Author  >>

PPDF>
```

Fig 12. Metadata of PDF Document

The process to discover these malicious documents could be forensic analysis or learning based detection.

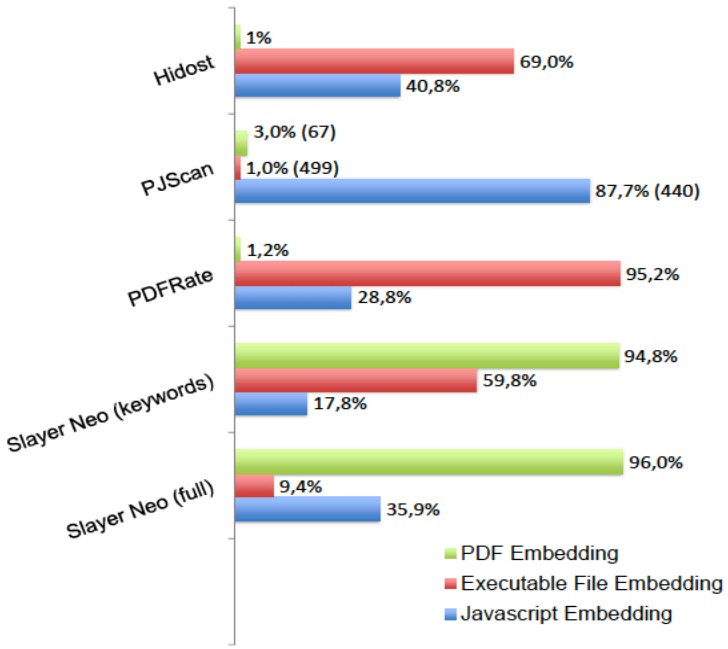

Fig 13. Detection rate of PDF malware

S. Pai in [5] talks about forensic analysis as a method where we can use machine learning techniques to identify the important aspects that could classify a document as malicious or by focusing on the source of the document. If the source was a spam email or phishing website, then the content can be analyzed. Further analysis would explore the actions performed by the malicious code and their consequences on the system or the data.

```
File: CVE-2010-0188_PDF_0B86453BCCEC6B52B98029C14C03C7D5
MD5: 0b86453bccec6b52b98029c14c03c7d5
SHA1: efa160836e9b1e78d48032ac7042aacfcc66e8e4
Size: 3506 bytes
Version: 1.3
Binary: False
Linearized: False
Encrypted: False
Updates: 0
Objects: 10
Streams: 1
Comments: 0
Errors: 0

Version 0:
        Catalog: No
        Info: No
        Objects (10): [1, 2, 3, 4, 6, 7, 8, 9, 10, 11]
        Streams (1): [8]
                Encoded (1): [8]
        Suspicious elements:
                /Names: [10]
                /JS: [11]
                /JavaScript: [11]
```

Fig 14. Analysis of PDF document

A. Damodaran in [6] talks about keyword-based analysis. This depends on finding the keyword objects in the document as */JavaScript* or */JS.* If the keyword is found, the analyst could further study the stream objects and decompress them to understand their real function. If there are no keywords found, then the document can be considered to be safe for the user to access.

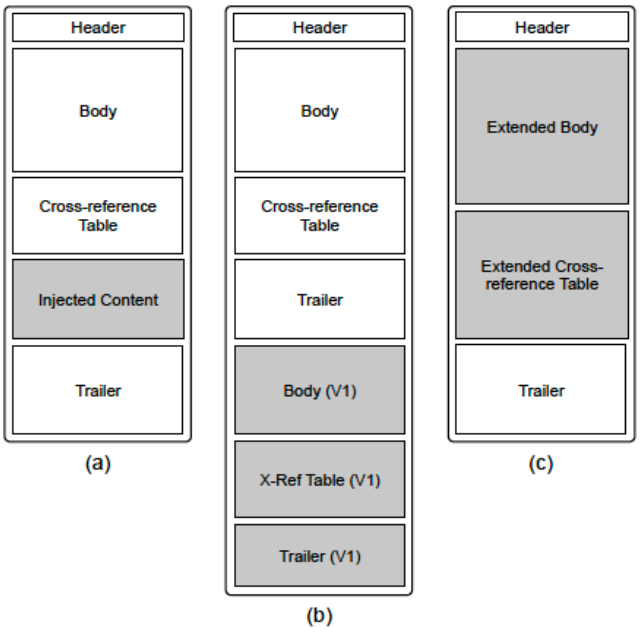

Fig 15. Injection of content in PDF files

Another method is Tree based analysis where a tree is created showing the interconnections amongst the objects of the PDF document. The system begins with the trailer object of the PDF document to find the */Root* keywords from where it starts building the tree [6]. The system then studies each object. Most of the malicious PDF files end with objects containing suspicious actions.

```
PPDF> tree

/Catalog (1)
        /Fields (6)
        array (8)
        /Pages (4)
                /Page (11)
                        /Pages (4)
                        stream (9)
                        /Action /Transparency (10)
                /ProcSet (12)
        /Outlines (3)
        dictionary (5)
        /JavaScript (7)
                /Names (15)
                        /Action /JavaScript (14)
                                stream (13)
/Info (2)

PPDF>
```

Fig 16. Tree Structure of PDF

Code based Analysis is based on studying the embedded code i.e. locating the /JavaScript code in the document and identifying what vulnerability it can exploit – known or unknown. The goal is to figure out the scripting code which are embedded into the files without getting into the internal depth of the files. This helps in finding if the lines of the code match to known vulnerabilities, and thus determine the maliciousness of the file.

```
PPDF> rawobject 7

<< /JavaScript 15 0 R >>

PPDF> rawobject 5

<< >>

PPDF> rawobject 3

<< /Type /Outlines
/Count 0 >>

PPDF> rawobject 1

<< /AcroForm 6 0 R
/Threads 8 0 R
/ViewerPreferences << /PageDirection /L2R >>
/OpenAction << /S /JavaScript
/JS (this.STARTSCRIPT\(\)) >>
/Pages 4 0 R
/Outlines 3 0 R
/Type /Catalog
/PageLayout /SinglePage
/Dests 5 0 R
/Names 7 0 R >>
```

Fig 17. Objects in PDF Document

Using the features found during the analysis of PDF files with over a dataset of 25000 files, a tool was created to check perform the task for identifying the features by reading the PDF file and identifying if it was malicious or not.

A PDF document also contains elements like JavaScript and URLs. To tackle the problem of harmful URLs, a ML model was used using logical regression. Logistic regression is a predictive analysis model which describes the data and draws inference relationship analysis for the binary attributes which depend on one or more nominal, ordinal, interval or ratio-level independent attributes. Implementation of this algorithm was done due to its property of handling multivariate dependent classes.

Performance of the model is evaluated using the confusion matrix to test various evaluation parameters as recall and area under ROC. Following are the results and visual representations of the obtained results.

Results:

```
True Positives: 10780

True Negative: 10835

False Positive: 1

False Negative: 664

Total: 22280


Accuracy: 97.0153

F-Score: 97.0078741

Recall: 94.1978

Precision: 99.9907
```

The performance results are computed using the following formulas:

Accuracy = (T P + T N) / T P + F N + F P + T N

F-Score = 2 ∗ (P ∗ R) / (P + R)

Recall = T P / (T P + F N)

Precision = T P / (T P + F P)

Evaluation Metric results obtained from the confusion matrix table show that the Accuracy and F-measure are the metrics which are used for the evaluation of the classifier performance. The F- measure is defined in terms of Recall (R) and Precision (P). If the evaluation

metrics have higher value, then the classifier is best suitable for the data set. The evaluation metrics results described effectively by the confusion matrix are:

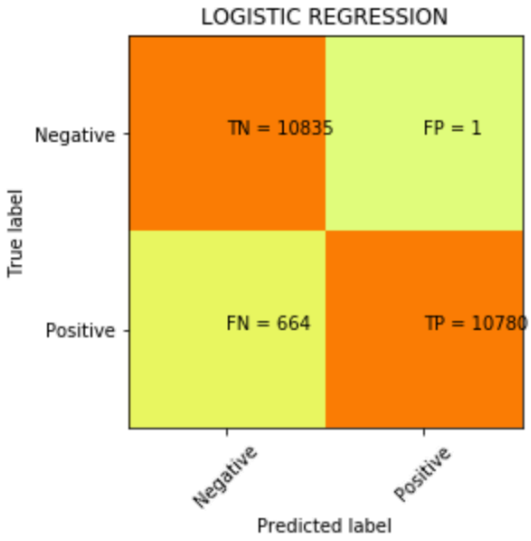

Fig 18. Confusion Matrix

The entries in the Fig. 18 of the confusion matrix have the following meaning in our context:

True Negative (TN): Malicious URL samples are labelled as a Malicious.

False Positive (FP): Malicious URL samples are labelled as a Legitimate(non-malicious).

False Negative (FN): Legitimate(non-malicious) URL samples are labelled as Malicious.

True Positive (TP): Legitimate(non-malicious) URL samples are labelled as a Legitimate(non-malicious).

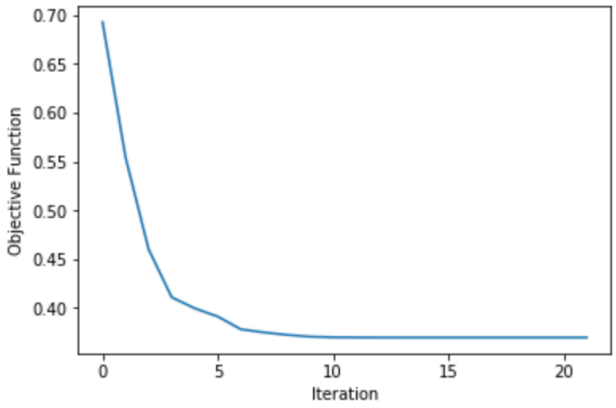

Fig 19. Convergence Curve

Figure 19 gives the Convergence curve of the objective function plotted against the number of iterations for the logistic regression method applied to URL dataset.

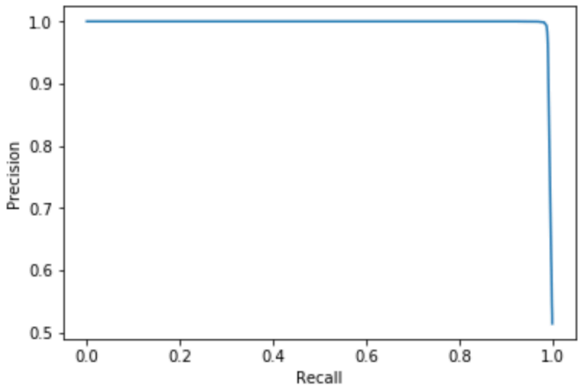

Fig 20. Precision Recall Curve

Figure 20. Gives the precision-recall curve which shows the relationship between the precision (positive prediction value) and recall (sensitivity) for every possible cut-off. The graph is compute as:

X-axis: Recall = sensitivity = TP / (TP + FN)

Y-axis: Precision = positive predicted value = TP / (TP + FP)

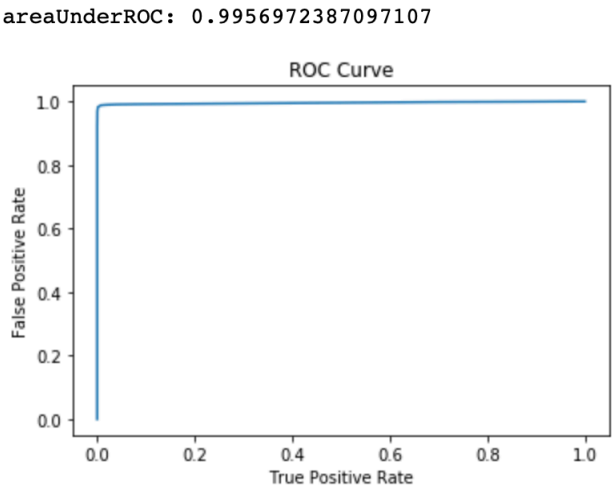

Fig 21. AUC-ROC

Figure 21 shows the AUC-ROC(area under curve - receiver operating characteristic curve) which graphically illustrates the diagnostic ability of the logistic regression model, which has varied discrimination threshold. The ROC curve is created by plotting the false positive rate (FPR) known as fall-out or calculated as 1-specificity, against the true positive rate(TPR) known as sensitivity or recall, at various threshold settings.

Python tool which analyses a file to determine the maliciousness of a file is implemented to give a verdict on a given input. This script reads through the content of a given PDF file and identifies the objects which are suspicious and gives the verdict based on the same.

The results from scanning a number of mixed, benign and malicious files is given below:

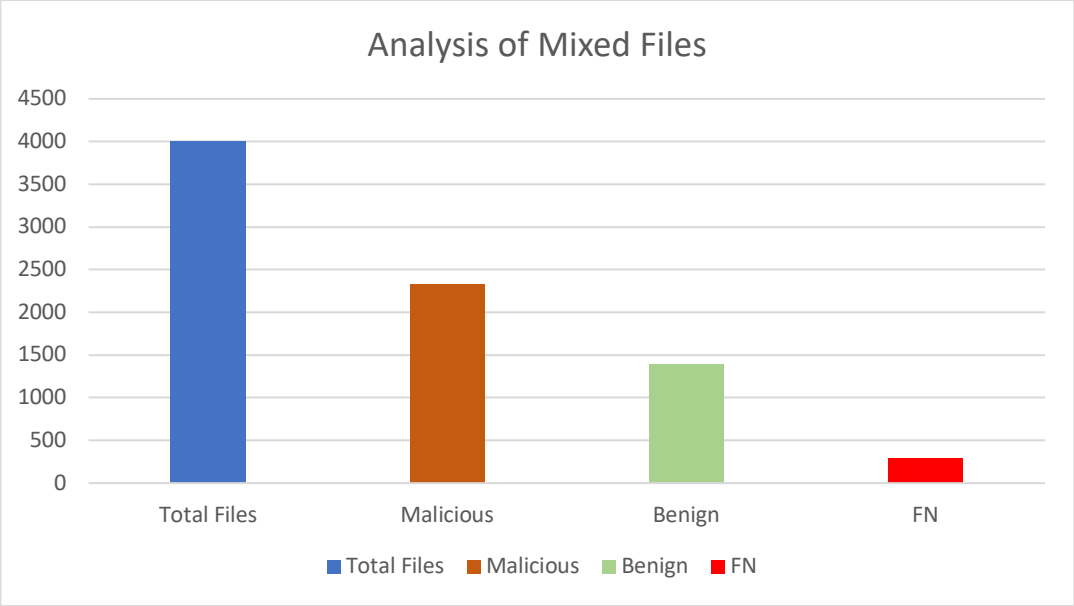

Fig 22. Analysis of Mixed Files

Thus, we can see that the tool is working with 92.92% accuracy on identifying the malicious files and 92.46% accuracy on identifying the benign files when the entire input set is a mixture of both malicious and benign files.

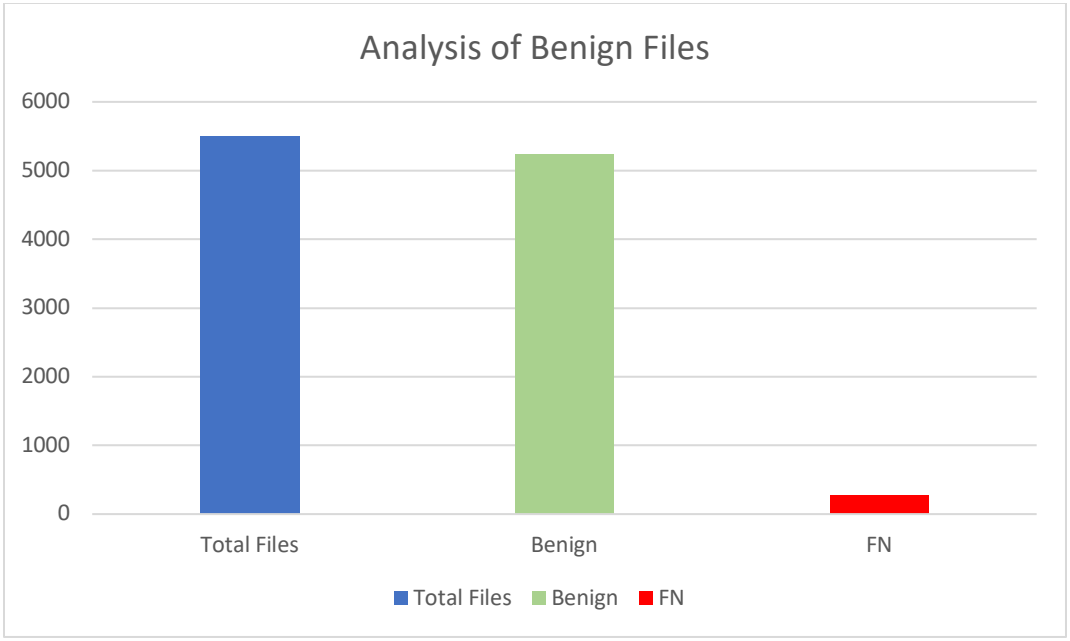

Fig 23. Analysis of Benign Files

Thus, we can see that the tool is working with 95.07% accuracy on identifying the benign files when the entire input set is of benign files.

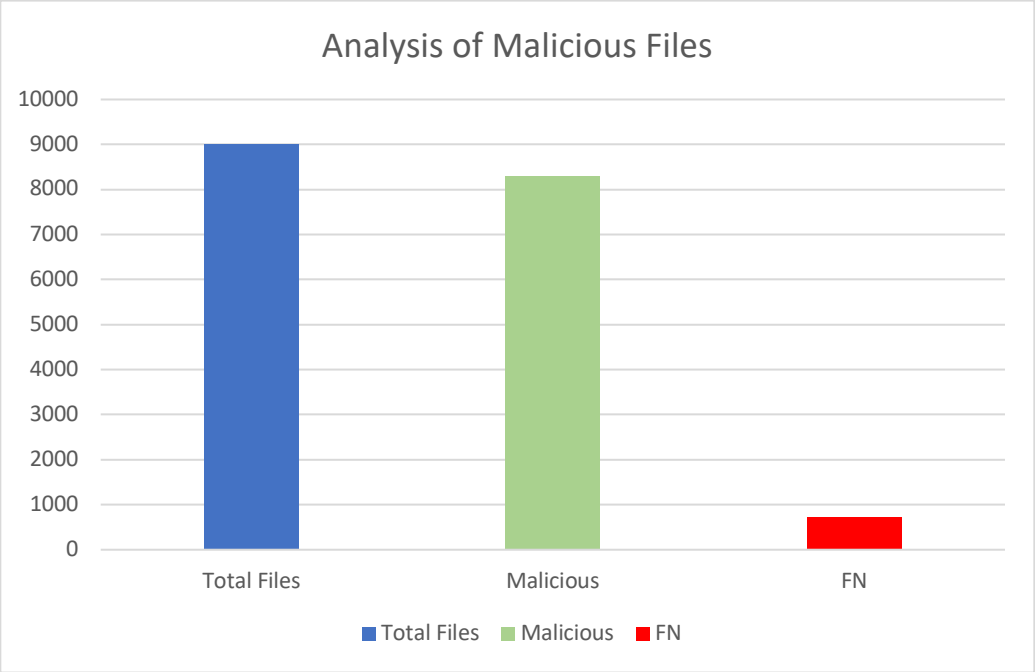

Fig 24. Analysis of Malicious Files

Thus, we can see that the tool is working with 92.07% accuracy on identifying the malicious files when the entire input set is of malicious files.

Chapter 5

Conclusion

Attackers have become stronger and economically capable to carry out attacks. These attacks are becoming more and more sophisticated and difficult to detect. Attackers learn the vulnerabilities in the system and figure out the loopholes in the system that are analyzing such attacks and therefore, it is becoming more and more important for the development of methods and techniques that will stop the malware attacks on the extensively used PDF documents.

A PDF document is more and more flexible to embed malwares into it. We studied the various functions which can help embed malware into PDF. Malicious URLs, JavaScript and ActionScript code are the most exploited features of the PDF document to embed the malware.

In this report, we have also looked at some examples of how malware is embedded into the document to take advantage of the vulnerabilities found in the reader system. We have also seen it with respect to the formation of the document giving us better insights. We have seen some methods that can be used to analyze and stop such attacks from happening such as forensic analysis and developing features to train models of machine learning.

However, attackers find loopholes in these techniques and are successful in obfuscating the actual embedded code to evade from such analysis findings. With the economic gains and usage of refined methods to carry out attacks, we need to provide security guarantees for the usage of PDF documents and thus require the development of algorithms which can help stop these attacks and prevent them. There are few vulnerabilities in the machine learning techniques which the attackers sooner or later will exploit. Thus, we need to have advancements in better machine learning techniques as well.

Chapter 6

Future Work

Learning based detection methods are dependent on using machine learning techniques. Over the years, the use of machine learning techniques has increased. It helps simplify the complex tasks of identifying the malware and stopping the attacks. Machine learning techniques are based on pre-processing and feature extraction [7]. In pre-processing, the contents of the document are extracted without executing the file in the reader. These contents are analyzed, and suspicious files are submitted for further analysis. Suspicious files are executed on virtual machines or sandboxes to monitor their behavior. This method helps to bypass any means that can be implemented to bypass initial analysis. In feature extraction [7] [8], the objects such as /JavaScript are given more focus and these contents are extracted from the document for examination. The structural features of the document such as the names of the objects used are also considered for analysis and extraction of any suspicious information found. Anusha Damodaran, et al., in [6] describe various ways in which the learning-based models can be implemented. Samir G. Sayed and Mohamed Shawkey in [8] give an example of their model which is based on the learning-based detection methods. They use a feature extraction, feature selection and classifier model for training their model and finding malicious PDF documents.

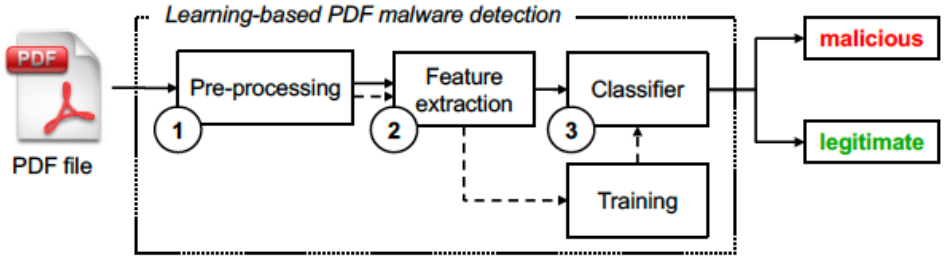

Fig 25. Learning-based detection architecture

Thus, by learning the techniques from this report and the tools, it can be re-engineered into features to train a model and improvise on the results that we get out of the same.

REFERENCES

[1]. C.Curtis, X.Hu, H.Yin,A.V.BhaskarandM.Zhang,“Extract Me If You Can: Abusing PDF Parsers in Malware Detectors,” in *23th Annual Network & Distributed System Security Symp.,* San Diego, California*, USA*, 2016.

[2]. W Xu, Y. Qi and D. Evans, “Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers,” in *23th Annual Network & Distributed System Security Symp., San Diego, California, USA*, 2016.

[3]. Maiorca, Davide, and Battista Biggio. "Digital Investigation of PDF Files: Unveiling Traces of Embedded Malware." [astro-ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field. July 17, 2017. Accessed November 10, 2018. https://arxiv.org/abs/1707.05102.

[4]. F. D. Troia, C. A. Visaggio, T. H. Austin and M. Stamp, "Advanced transcriptase for JavaScript malware," *2016 11th International Conference on Malicious and Unwanted Software (MALWARE)*, Fajardo, 2016, pp. 1-8. doi: 10.1109/MALWARE.2016.7888737

[5]. S. Pai, F. Di Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, "Clustering for malware classification," J. Comput. Virol. Hacking Tech., vol. 13, no. 2, pp. 95--107, 2017.

[6]. A. Damodaran, F. Di Troia, C. A. Visaggio, T. H. Austin, M. Stamp, "A comparison of static dynamic and hybrid analysis for malware detection", *J.C.V.H.T.*, pp. 1-12, 2015.

[7]. Isabelle Guyon, André Elisseeff, An introduction to variable and feature selection, *The Journal of Machine Learning Research*, 3, 3/1/2003

[8]. Daniele Ucci, Leonardo Aniello, Roberto Baldoni, "Survey on the Usage of Machine Learning techniques for Malware Analysis", ACM Transactions on the Web, October 2017.

[9]. Sayed, Samir G. and Mohmed Shawkey. "Data Mining Based Strategy for Detecting Malicious PDF Files." *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)* (2018): 661-667.

[10]. S. H. T. Mavric and C. K. Yeo, "Online binary visualization for Pdf documents," *2018 International Symposium on Consumer Technologies (ISCT)*, St. Petersburg, 2018, pp. 18-21. doi: 10.1109/ISCE.2018.8408906

[11]. https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the_rise_of_pdf_malware.pdf

[12]. Adobe, “PDF Reference and Adobe Extensions to The PDF specification,” https://www.adobe.com/devnet/pdf/ pdf reference.html/, accessed: 2018-03.

[13]. Hyrum S. Anderson, Anant Kharkar, Bobby Filar and Phil Roth, “Evading Machine Learning Malware Detection”, Black Hat USA 2017, July 22-27, 2017, Las Vegas, NV, USA

[14]. Himanshu Pareek, P R L Eswari and N. Sarat Chandra Babu, “Malicious PDF Document Detection Based on Feature Extraction and Entropy”, *International Journal of Security, Privacy and Trust Management (IJSPTM),* Vol 2, No 5, October 2013

[15]. Maiorca, D., Giacinto, G., Corona, I.: A pattern recognition system for malicious pdf files detection. In: MLDM, pp. 510–524 (2012)

[16]. N. Srndic and P. Laskov, “Detection of Malicious PDF Files Based on Hierarchical Document Structure,” in Proceedings of the 20th Annual Network & Distributed System Security Symposium, San Diego, CA USA, 2013

[17]. B. Cuan, A. Damien, C. Delaplace, and M. Valois, “Malware Detection in PDF Files Using Machine Learning,” REDOCS, Tech. Rep. Rapport LAAS No. 18030, Feb. 2018.

[18]. J. S. Cross and M. A. Munson, “Deep pdf parsing to extract features for detecting embedded malware,” Sandia National Laboratories, Albuquerque, New Mexico 87185 and Livermore, CA 94550, Tech. Rep. SAND2011-7982, Sep. 2011.

[19]. T. Mitchell, Machine Learning. McGraw Hill, 1997.

[20]. J. Zhang and J. Rabaiotti, “The PDF Exploit: Same Crime, Different Face,” https://www.symantec.com/ connect/blogs/pdf-exploit-same-crime-different-face/, accessed: 2018-03

[21]. J. Zhang, “Make “Invisible” Visible - Case Studies in PDF Malware,” in Proceedings of Hacktivity 2015, Budapest, Hungary, 2015.

[22]. Choi, Y.H. et al.: Toward extracting malware features for classification using static and dynamic analysis. Computer Netw. Technology. (ICCNT) (2012)

[23]. D. Maiorca, D. Ariu, I. Corona, and G. Giacinto, "A structural and content-based approach for a precise and robust detection of malicious PDF files," in 1st Int'l Conf. Information Systems Security and Privacy, 2015, pp. 27–36.

[24]. M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious Javascript code," in 19th Int'l Conf. World Wide Web, 2010.

[25]. D. Maiorca, I. Corona, and G. Giacinto, "Looking at the bag is not enough to find the bomb: An evasion of structural methods for malicious PDF files detection," in 8th ACM SIGSAC Symp. On Information, Computer and Communications Security, 2013.

[26]. C. Curtis, X. Hu, H. Yin, A.V. Bhaskar and M. Zhang, "Extract Me If You Can: Abusing PDF Parsers in Malware Detectors," in 23th Annual Network & Distributed System Security Symp., San Diego, California, USA, 2016.

[27]. https://blog.didierstevens.com/2008/05/19/pdf-stream-objects/ [05/05/2019]

[28]. http://www.simpopdf.com/resource/pdf-file-structure.html [05/05/2019]

[29]. https://resources.infosecinstitute.com/pdf-file-format-basic-structure/#gref [05/05/2019]